

**Original citation:**

Zheng, Y., Kerbyson, D. J. and Nudd, G. R. (1992) Efficient load balancing techniques for image analysis on an M-SIMD machine. University of Warwick. Department of Computer Science. (Department of Computer Science Research Report). (Unpublished) CS-RR-214

**Permanent WRAP url:**

<http://wrap.warwick.ac.uk/60903>

**Copyright and reuse:**

The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions. Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

**A note on versions:**

The version presented in WRAP is the published version or, version of record, and may be cited as it appears here. For more information, please contact the WRAP Team at: [publications@warwick.ac.uk](mailto:publications@warwick.ac.uk)



<http://wrap.warwick.ac.uk/>

# Research Report 214

## Efficient Load Balancing Techniques for Image Analysis on an M-SIMD Machine

Zheng Y\*, Kerbyson DJ and Nudd GR

RR214

The computational requirements for the real time processing of image sequences is sufficiently high that some form of parallel hardware is essential. In the analysis of a sequence of images the areas of interest are moving objects which usually occupy only small distinct areas within the full field of view. A single instruction multiple data (SIMD) machine has considerable advantages for these types of operations where there is a high requirement for data parallel processing. However, on conventional SIMD machines, only the processors to which the moving objects are mapped onto have significant work-load. The remaining processors are idle during most of the processing period resulting in significant load imbalance and poor utilisation.

We describe here load balancing techniques for a Multiple-SIMD (M-SIMD) machine, consisting of a number of small conventional SIMD arrays (patches) connected together to form a larger M-SIMD array. Each SIMD patch can perform independent computations. Using the M-SIMD configuration idle processors can be re-allocated to process active regions of other images from an image sequence or from multiple sensors, significantly increasing the throughput and flexibility of the system. A 'voting' algorithm is presented for the calculation of the minimum number of patches the object is mapped onto along with a heuristic (near optimum) patch allocation process.

Key words: Load balancing, Image analysis, Multiple-SIMD architecture

\*Department of Computer Engineering, Shanghai University of Technology, Yangchang Road, Shanghai, 200072, CHINA.

# Efficient Load Balancing Techniques for Image Analysis on an M-SIMD Machine

Yanheng Zheng<sup>1</sup>, Darren J. Kerbyson, Graham R. Nudd

*VLSI Architectures Group,  
Department of Computer Science,  
University of Warwick,  
Coventry CV4 7AL,  
UK*

## Abstract

The computational requirements for the real time processing of image sequences is sufficiently high that some form of parallel hardware is essential. In the analysis of a sequence of images the areas of interest are moving objects which usually occupy only small distinct areas within the full field of view. A single instruction multiple data (SIMD) machine has considerable advantages for these types of operations where there is a high requirement for data parallel processing. However, on conventional SIMD machines, only the processors to which the moving objects are mapped onto have significant work-load. The remaining processors are idle during most of the processing period resulting in significant load imbalance and poor utilisation.

We describe here load balancing techniques for a Multiple-SIMD (M-SIMD) machine, consisting of a number of small conventional SIMD arrays (patches) connected together to form a larger M-SIMD array. Each SIMD patch can perform independent computations. Using the M-SIMD configuration idle processors can be re-allocated to process active regions of other images from an image sequence or from multiple sensors, significantly increasing the throughput and flexibility of the system. A 'voting' algorithm is presented for the calculation of the minimum number of patches the object is mapped onto along with a heuristic (near optimum) patch allocation process.

Key words: Load balancing, Image analysis, Multiple-SIMD architecture

## 1. INTRODUCTION

The computational requirements for the real time processing of image sequences is sufficiently high that some form of parallel hardware is essential [1, 2]. Research has been undertaken examining the performance of different parallel computing structures to meet these requirements, e.g. [3, 4, 5, 6]. One typical requirement is to process moving objects within an image sequence which first are recognised and then tracked [7, 8]. However, to utilise a massively parallel machine effectively, strategies have to be developed which take into account the variation in object densities within the field of view. In this paper we address one such strategy which has application to the generic problem of efficiently utilising massively parallel hardware.

A sequence of images typically contains several moving objects occupying only small distinct areas within the full field of view. Examples include; flight path analysis, automatic recognition and tracking of road vehicles. A Single Instruction Multiple Data (SIMD) architecture has considerable advantages for these types of operations where there is a high requirement for data

---

<sup>1</sup> The author was visiting the Department of Computer Science, University of Warwick, U.K. in 1991. He is permanently with the Department of Computer Engineering, Shanghai University of Technology, Yanchang Road, Shanghai, 200072, China.

parallel processing. The case of a processor array equal in size to that of the image is considered such that each image pixel is mapped to a separate processing element (PE). However, on conventional SIMD machines, only the processors to which the objects of interest are mapped onto have significant work-load. The remaining processors are idle during most of the processing period resulting in significant load imbalance and poor utilisation.

A Multiple-SIMD (M-SIMD) architecture, consisting of a number of smaller conventional SIMD arrays (patches) connected together to form a larger SIMD array, can perform the same computations as a conventional SIMD architecture but also allows the re-allocation of idle patches of the processor array. One such M-SIMD architecture familiar to the authors is the Warwick Pyramid Machine [3, 4]. It consists of a set of  $16 \times 16$  SIMD patches, each with an associated controller and an MIMD processor. These units are four way connected at all levels (between SIMDs, controllers and MIMDs) to form a scalable M-SIMD machine. Each SIMD patch can operate autonomously or in synchronization with other patches. When an image is mapped spatially across the set of SIMD patches, those patches containing no objects of interest can be allocated to some other task, or in the case considered here to active regions from other images, significantly increasing the throughput and flexibility of the system. The size of the SIMD patch is generalised, for the following load balancing techniques, to be  $n \times n$  PEs and the number of SIMD patches within the M-SIMD machine to be  $N \times N$ , with the patch or PE of coordinate (0,0) being in the bottom right corner of the array.

Load balancing on a distributed memory machine has been actively examined in recent years. Jeng and Seigel [9] have done much work on dynamic partitioning of large-scale computers to process application's with different computation structures and different degrees of parallelism while alleviating the fragmentation problem. Li and Cheng [10] are working on job scheduling in partitionable mesh connected system to increase performance.

The load balancing model used here has some unique features differing from the existing models. These are:

- a) The load pattern is deterministic. This is due to the image of the moving objects not changing very quickly. We assume that the PEs corresponding to moving objects on the next frame will remain similar as in the current frame.
- b) The load pattern can be more complicated than a rectangle (assumed in most of the recent papers).
- c) The processor allocation is very easily implemented using local communications such that all the pixels on a frame are shifted the same distance in both horizontal and vertical directions.
- d) Real time operation is required hence the time for any load balancing calculations are limited.

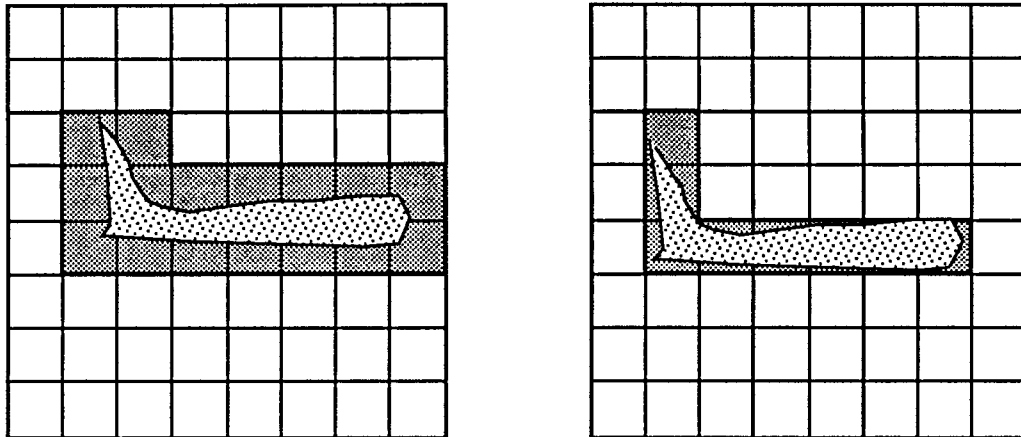
Three important aspects for the effective load balancing of an image analysis application on an M-SIMD machine are:

- 1) The extraction of image regions containing the moving objects from an image sequence. Various techniques are available for this process such as [11, 12, 13]. We assume that a suitable method is employed and results in the identification of the object areas.
- 2) The development of an 'voting' algorithm for calculating the optimum processor patch area to contain the objects.
- 3) The development of a heuristic allocation process to provide near optimum mapping.

The calculation of the optimum object area is described in section 2 and the heuristic allocation process in section 3. The improvements in throughput that can be achieved are discussed in section 4.

## 2. CALCULATION OF THE OPTIMUM OBJECT AREA

For efficient processing on a fixed computation topology, each object area must be optimized such that the number of occupied processors for any object size is kept to a minimum. For example an extracted area of an airplane is shown in Figure 1. Each of the smaller squares represents an SIMD patch joined together forming an 8\*8 patch M-SIMD machine. The number of occupied SIMD patches is reduced to 8 in the optimized case against 16 in the original. The benefits of this optimization are two fold: a greater number of idle SIMD patches can be used for the processing of other objects, and the communication overhead in the active areas is reduced.



(a) Original area  
(b) Area after optimization  
*Figure 1 - The extracted area of an airplane on an M-SIMD machine*

The minimum number of SIMD patches used for processing such an object can be found by the naive method of repeatedly shifting the object area and performing a count of the number of occupied SIMD patches in each shift. The shift occupying the least number of patches would be chosen. However a total of  $n^2$  shifts are required which is computationally expensive. A vote algorithm, with much lower computational overhead is described below.

The vote algorithm consists of three stages -

- 1) The first is to fill any gaps which occur within the extracted object either internally or concavities that occur on its boundary which are less than  $n$  in diameter. The filling results in unoccupied areas around or within the extracted object which cannot be used to free a whole SIMD patch being marked as occupied, simplifying the voting mechanism.
- 2) Each SIMD patch calculates a set of shifts that can be performed on the object area contained within it, to free its patch of the object. This is done for each of eight directions - North, South, East, West and each of the diagonals (only if such a shift exists).
- 3) The shifts are broadcast to the other SIMD patches allowing them to vote on the effects that each shift would produce. A count of the number of patches that would be freed and the number of new patches which would be occupied is performed. The difference between the two is the effective number of SIMD patches freed, the maximum of which represents the optimum mapping of the object across the M-SIMD array.

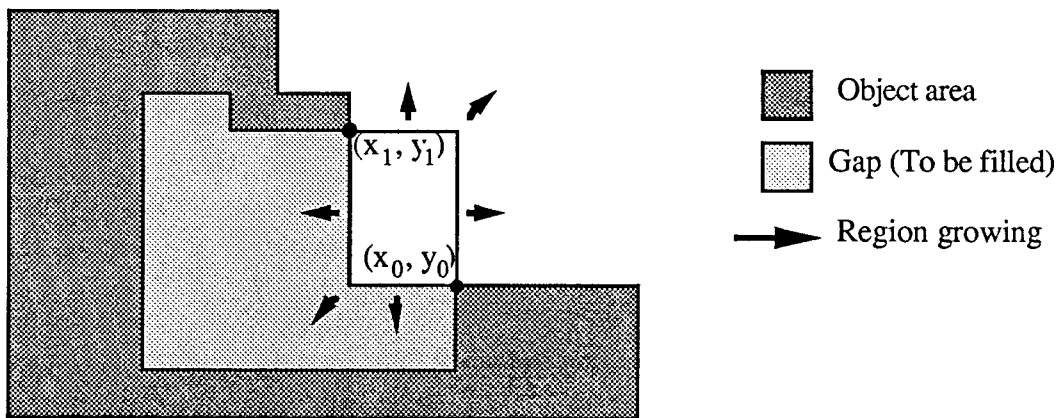
The vote algorithm is designed to execute on an M-SIMD machine. Most of the operations for the calculation are taken from neighbouring SIMD patches keeping the required communication small. The gap filling procedure and the voting mechanism are described below.

## 2.1 Gap filling of the extracted object area

The SIMD processors containing an object pixel is labelled with a '1' (an occupied PE) as a result of the object detection algorithm and a '0' otherwise (an unoccupied PE). The gap filling algorithm can be thought of as changing the state of any PE from '0' to '1' if and only if such changes do not produce any side effects on the number of SIMD patches occupied by the object. This is achieved by setting unoccupied PEs to be occupied if they lie between two occupied PEs separated by a distance  $< n$  PEs either within a horizontal or vertical line, or both horizontally and vertically. The unoccupied PEs which are set with this operation could never result in a whole SIMD patch being freed and does not effect the number of SIMD patches used.

The case of occupied PEs separated by a distance  $< n$  vertically is found by shifting southwards a one-bit mask from each of the objects southern edge PEs a distance of  $n$ , and marking any overlap between the shifted mask and the object area. These marked locations are then shifted northwards, setting each PE visited to be occupied until a similar overlap between the shifted mask and the object is found. This is very easily performed on an M-SIMD machine when operating as a conventional SIMD machine. A similar operation to the east can be performed.

The second case of occupied PEs separated by a distance  $< n$  both horizontally and vertically is slightly more complicated, an example of which is shown in Figure 2 for the PEs  $(x_0, y_0)$  and  $(x_1, y_1)$ . These two points could be located within a single SIMD patch after a shift operation. A check for this case can be made by marking each object corner PE and propagating the resulting mask to fill an  $N \times N$  square of PEs centred around each PE, and checking for overlapping of the mask from other object corner PEs.



*Figure 2 - Filling gaps between unconnected occupied PEs*

The PEs representing the 'Manhattan path' between any two corner points, within the  $n \times n$  square, are set as shown in Figure 2 and a region growing algorithm is performed to fill the gap. Note however that the 'Manhattan path' can be marked in one of two ways, either going horizontally before vertically or vice - versa, forming a rectangle. Both of these possibilities need to be considered by the region growing operation and are shown in Figure 2. Each rectangle side is used to start the region growing operation. The side finishing the growing operation first is assumed to be the internal gap (which is now filled). The other side of the rectangle would be filling the part of the image external to the object requiring far more iterations of the region growing operation. Multiple gaps are also dealt with by this method.

For the object of Figure 2, the region growing from the lower and left sides of the rectangle would finish first. The filing of this area does not increase the number of occupied SIMD patches by the object. After the gap filling operation, every pair of occupied object PEs which could be located in the same SIMD patch after a shift operation are linked with the PEs inbetween in the horizontal or vertical directions marked as being occupied i.e. treated from now-on as part of the object area.

## 2.2 Calculation of image shifts which will free an SIMD patch.

An SIMD patch may be either partially or fully covered with the object area as can be seen in Figure 1. The patches which are only partially covered by the object may be transformed to a free patch, available for the processing of another object, by a suitable shift of the object in one of 8 directions. The calculation of the shift required to free an SIMD patch with a simple shift is as follows.

Each object boundary PE is labelled as being on the North, if the PE is on the northern boundary of the object and is in the bottom half of the SIMD patch (or South if the PE is on the southern boundary of the object and is in the top half of the PE) and so on for east and west. Similarly concave corner PEs are labelled their respective corners if for instance a NE boundary PE is in the SW quadrant of the SIMD patch. Now consider the north object edge PEs, the maximum of the northern edge PEs, (i.e. the top of the object within the bottom half of the SIMD patch) represents the required shift south to clear the SIMD patch. Similar operations are repeated for each of the remaining directions.

The result of these calculations is a table of values representing the shifts (if any) which will move the object out of the respective SIMD patch boundary. The calculations of these shifts can be performed on each of the SIMD patches in parallel using their M-SIMD operational capability. Each of the shifts are then broadcast to the other SIMD patches covered by the object so each may vote on whether the shift will free itself as well (or not).

## 2.3 Voting on an image shift

Each SIMD patch receives a requested shift ( $S_x, S_y$ ), from the list of calculated shifts within each of the other object patches, where  $S_x$  and  $S_y$  represent the requested shifts in the horizontal and vertical respectively (both non-zero for a diagonal shift). Each object patch then votes on this - either a *YES*, *PASS* or *NO*;

a vote *YES* means that the shift will free the object patch

a vote *PASS* means that the shift neither frees the object patch or occupies other patches

a vote *NO* means that the shift will occupy other patches as well as itself

For the following vote algorithm a notation of '0' and '1' is used to denote the state of an unoccupied and occupied SIMD patch respectively, and a subscript of '-' and '+' to denote the situation before and after the requested shift. Each SIMD patch is addressed as (X, Y) within the M-SIMD machine. The algorithm is illustrated for a requested shift in the North-East direction (the algorithm may easily be changed for other directions).

The patch (X,Y) could be occupied by part of the object shifted from the patches (X,Y-1), (X-1,Y) and (X-1,Y-1) for the North-East shift. The possible occupancy can easily be found by using a masking operation and the associative response within each of these neighbouring patches. If a neighbouring patch contains part of the object which will occupy the patch (X, Y) after the shift, a 1<sub>+</sub> is sent to the patch (X, Y) otherwise a 0<sub>+</sub> is sent. The masking required is:

For patch (X,Y-1) If PE(a,b) is occupied send a '1<sub>+</sub>' where  $(1 \leq a \leq (n-S_x) \ \& \ (n-S_y) \leq b \leq n)$

For patch (X-1,Y) If PE(a,b) is occupied send a '1<sub>+</sub>' where  $((n-S_x) \leq a \leq n \ \& \ 1 \leq b \leq (n-S_y))$

For patch (X-1,Y-1) If PE(a,b) is occupied send a '1<sub>+</sub>' where  $((n-S_x) \leq a \leq n \ \& \ (n-S_y) \leq b \leq n)$

The patch (X,Y) votes dependent upon the values received from its neighbours and its own occupied status according to the following:

If the patch had status '0.' and only '0<sub>+</sub>' are received then vote *PASS*

If the patch had status '1.' and a '1<sub>+</sub>' is received then vote *PASS*

If the patch had status '0.' and a '1<sub>+</sub>' is received then vote *NO*

If the patch had status '1.' and a '0+' is received and at least one of the requested shifts ( $S_x$ ,  $S_y$ ) is contained within one of the calculated shifts for the patch (from section 2.2) then vote *YES*

The votes are calculated for each patch in parallel using their M-SIMD capability. The votes from each patch are then sent to the patch requesting the shift for accumulation. The difference between the *YES* and *NO* votes is the effective number of patches freed on that requested shift, the maximum of which (over all requested shifts) gives the shift for optimum object mapping which can then be used to shift the object data.

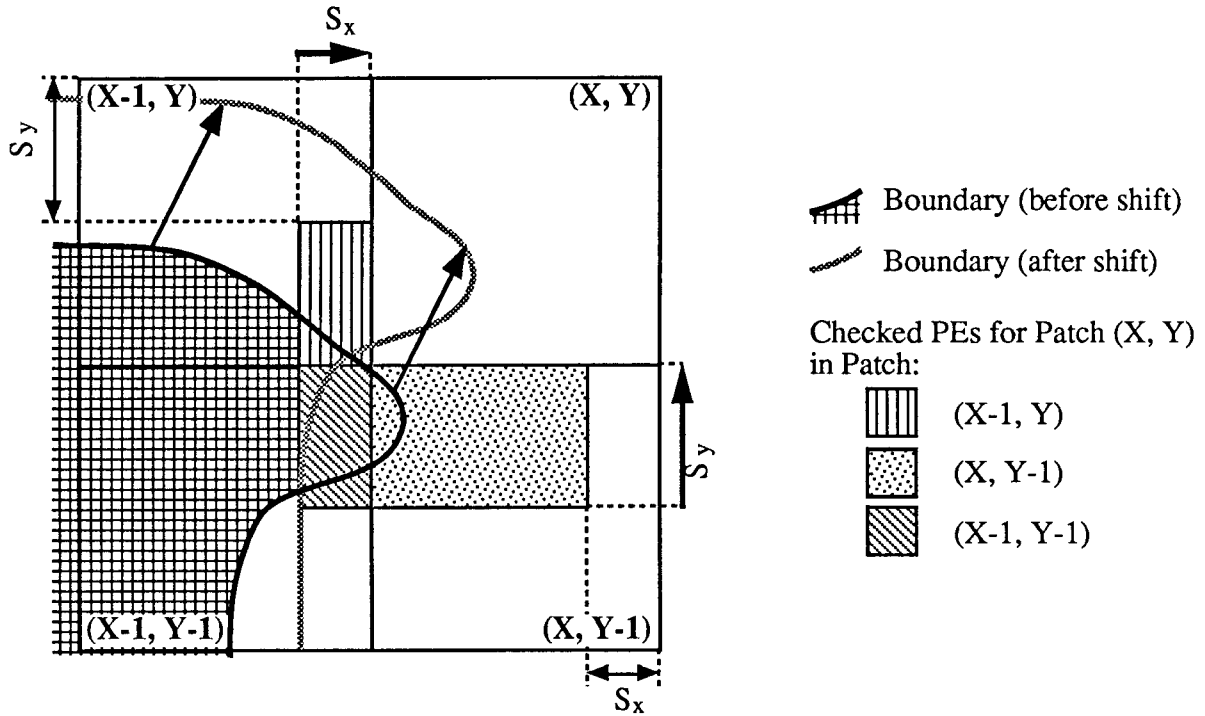


Figure 3 - Example of a North-East shift showing checked PEs for the North-East Patch

An example is shown in Figure 3 with part of an object mapped across three patches. The shift shown is in a North-East direction with  $S_y > S_x$ . Before the shift three patches are occupied and after the shift patch (X, Y-1) becomes freed but patch (X, Y) becomes occupied. These two patches vote *YES* and *NO* respectively, as shown in Table 1. The shift results in an equal number of *YES* and *NO* votes for the part of the object shown - there is no change in the number of occupied patches.

Patch	Occupied status		Vote
	Before	After	
(X, Y)	0.	1+	<i>NO</i>
(X-1, Y)	1.	1+	<i>PASS</i>
(X, Y-1)	1.	0+	<i>YES</i>
(X-1, Y-1)	1.	1+	<i>PASS</i>

Table 1 - Voting of the patches shown in Figure 3.

The main requirements of the vote algorithm is involved in the broadcasting of the requested shifts and in receiving the vote results. However only the patches which can be freed after a shift produce such requests. The overhead of the algorithm is negligible compared with the available processing time within each frame period.



### 3. OBJECT PROCESSOR ALLOCATION

Once the area of the object on a frame is extracted it may be moved to a spare set of processor patches for further processing. Two strategies can be adopted for this mapping; a first fit allocation strategy and a look ahead allocation strategy. The First-fit allocation strategy finds a set of free patches which satisfies the requirement of the object area and allocates them to the object. However this can result in poor utilisation, with a fragmentation problem when further object areas are mapped across the remaining patches. The Look- ahead allocation strategy uses the object area information to best allocate further objects (assuming that the objects are slowly varying in size). The latter allocation strategy results in greater utilisation.

The extracted areas are usually irregular in shape and the best allocation strategy is difficult to determine theoretically - simulation is required. Two simulation methods have been considered here - an exhaustive and a heuristic method.

#### 3.1 Exhaustive method

The exhaustive method tests all possible processor allocations to the objects and selects the one in which the number of allocated objects is a maximum (the optimum allocation). The algorithm is composed of:

- \* The first object area is assigned arbitrarily.
- \* For the  $i^{\text{th}}$  area ( $i \geq 1$ ), calculate all of the  $(i^{\text{th}}+1)$  possible assignments which do not conflict with previously allocated patches.
- \* Repeat above until no further assignments are possible.
- \* Select the best allocation pattern - the one with greatest number of object allocations.

Although this algorithm obtains the optimum processor allocation, it is time consuming. In the worst case (when the object area occupying only one patch), the computing complexity is  $(N^2)!$  where  $N^2$  is the number of SIMD patches in the M-SIMD machine (for the WPM,  $N = 8$ ). This complexity makes it unusable within a real-time environment.

#### 3.2 Heuristic method

The heuristic method is designed to reach near optimum allocation but with a linear complexity and is described in the following steps.

- 1) The first object area is assigned arbitrarily
- 2) The second area should be assigned such that it is connected with the first one. All allocations which satisfy this condition are candidate assignments for the second area, one of which is part of the best allocation possible.

The computing complexity of the step is  $O(m)$ , where  $m$  is the number of the processors in the rectangle around the first object area composed of the four edges:

$$\begin{aligned} X &= (X_{\min} - 1) \bmod n, & X &= (2X_{\max} - X_{\min} + 1) \bmod n, \\ Y &= (Y_{\min} - 1) \bmod n, & Y &= (2Y_{\max} - Y_{\min} + 1) \bmod n \end{aligned}$$

where  $X_{\min}$ ,  $X_{\max}$ ,  $Y_{\min}$  and  $Y_{\max}$  are the minimum and the maximum patches containing the object area horizontally and vertically respectively and  $(n*n)$  is the number of patches within the M-SIMD array. No candidate assignments are available beyond this area. The modulus term is a result of the torus network of the M-SIMD machine.

If a candidate assignment is found at position  $P(X_2, Y_2)$  then it can be shown that a further assignment can be made at position  $P(-X_2, -Y_2)$ . This results in the number of possible second area allocation candidates reduced by one half reducing the search area.

3) The third and subsequent areas are assigned for each of the second candidate assignments with a position of  $P(X_i, Y_i) = P((i-1)X_1 \bmod n, (i-1)Y_1 \bmod n)$  where  $i \geq 3$ . This continues until a conflict occurs between the next assignment and previously assigned patches. It can be thought of as forming an extension line of constant gradient originating from the first object area. For example,

- if  $(X_2 = X_1)$  then the extension line will be vertical, conflicting after one cycle of the array (due to the M-SIMD torus network).
- if  $(Y_2 = Y_1)$  then the extension line will be horizontal, conflicting after one cycle
- if  $((X_2 - X_1) = (Y_2 - Y_1))$  then the extension line will be at a slope of  $\pi/4$ , conflicting after one or more cycles.

If the horizontal shift is small e.g.  $(X_2 \approx X_1)$  the extension conflicts after one cycle, otherwise, it continues further. An example extension trace is shown in Figure 4 with the allocations of the first five areas. Note the wrap-around effect of the torus network in the third area. The extension ends only by conflicting with the first assigned area.

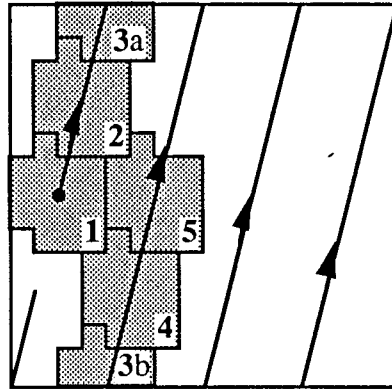


Figure 4 - The extension trace in the situation  $(x_2 < y_2)$

The extension line is extended for further cycles across the array by a respective shift of the line horizontally (when the slope is  $> \pi/4$ ) or vertically (when the slope is  $< \pi/4$ ) after a conflict occurs. If no such continuation is possible then no spare patches exist to process further objects.

An extension line is used for each of the candidate second assignments. The extension line which produces the maximum number of allocations is the one used for the second and subsequent allocations of the object across the M-SIMD patches.

### 3.3 Implementation of the Heuristic method

Simulation results from the Heuristic method have produced results which approach the optimum number of object allocations found by the exhaustive method. The performance of regular shaped objects using the heuristic method typically achieves 100% of the exhaustive method and irregular shaped objects typically achieve 80%.

The largest computing time of the patch allocation happens at following situation: both of  $(X_{\max} - X_{\min})$  and  $(Y_{\max} - Y_{\min})$  are  $\lfloor N/2 \rfloor$ . The computation time for assigning the second area increases with the search area, and the image size. A rectangular image would produce conflicts after fewer iterations with the extension line method resulting in decreased computation time. The computation time required for the third and subsequent object area assignments is heavily depended on the number of successful second assignments.

The calculation of the heuristic allocation method can be performed using binary data - suitable for efficient implementation within a single SIMD patch. If the number of SIMD PEs within the patch ( $n \times n$ ) is equal to the number of patches within the M-SIMD machine ( $N \times N$ ) then each PE is used to represent the allocation of each patch, using the torus network within the SIMD patch to represent the torus network of the whole M-SIMD machine. If these two sizes vary, a suitable mapping technique can be employed resulting in slight communication overheads.

#### 4. PERFORMANCE OF THE LOAD BALANCING

The time taken to perform the calculation of the object minimum patch mapping (section 2) along with the processor allocation method (section 3) is shown in Table 2 for a typical object of size  $4 \times 4$  patches. Note that both of these operations use binary data and so are very efficiently mapped onto an SIMD array. Table 2 also shows the maximum and average increase in throughput that can be obtained using these techniques. The times given are on the prototype Warwick Pyramid machine (WPM) [3,4] which is implemented using the AMT DAP, a bit-serial SIMD processor operating at a speed of 10 MHz.

	Operation	Time ( $\mu$ s)	Increase in throughput
Object patch mapping	Gap filling	40	4 (Maximum) 2 (Average)
	Voting	4	
	Result accumulation	24	
	Calc. of best result	15	
	Image shift	24	
	Total	107	
Patch allocation	Calc. requested shifts	12	$N^2/A$ (Maximum) $0.75 N^2/A$ (Average)
	Calc. allocation strategy	56	
	Image shift	48*	
	Total	116	
	Total Total	223	

A = the number of occupied patches  
\* per patch shifted across

Table 2 - Performance of the load balancing techniques of section 2 and 3.

The time taken for the patch allocation increases with the distance the object is to be shifted. The time of 48  $\mu$ s in Table 2 assumes the object is only shifted one full patch within the WPM and scales linearly with the number of patches traversed. However the time taken to perform the load balancing is small in comparison to the frame period (typically 1% of the frame period), and the throughput of the M-SIMD machine can increase by a factor of  $2 \times (0.75N^2/A)$  where  $N^2$  is the number of SIMD patches and A is the number of occupied patches.

#### 5. CONCLUSION

For the processing of image sequences, the configuration of an M-SIMD machine can dramatically increase the flexibility and throughput of the whole system. The results presented here have shown that the throughput within an M-SIMD machine could be increased by a factor of  $1.5 \times (N^2 / A)$ , where  $N^2$  is the number of PEs in an SIMD patch, and A is the object areas (in terms of whole patches). This increase is achieved by load-balancing techniques requiring the processing of binary data which can be efficiently and quickly performed on an M-SIMD array.

## ACKNOWLEDGEMENT

This work is supported in part by the U.S. Innovative Science and Technology grants N00014-87-G-0241 and N000014-90-J-1919 administered by Dr K. Bromley of the Office of Naval Research.

## REFERENCES

- 1 R.Cypher and J.L.C.Sanz, "SIMD Architectures and Algorithms for Image Processing and Computer Vision", IEEE Trans. on ASSP., Vol. 37(12), Dec. 1989.
- 2 M.Maresca, M.A.Lavin and H.Li, "Parallel Architecture for Vision", Proceedings of the IEEE, Vol. 76(8), Aug. 1988.
- 3 G.R.Nudd, D.J.Kerbyson, T.J.Atherton, N.D.Francis, R.A.Packwood and G.J.Vaudin, "A Massively Parallel Heterogeneous VLSI Architecture for MSIMD Processing", in Algorithms and Parallel VLSI Architectures, Vol. B, Elsevier North Holland, 1991.
- 4 G.R.Nudd, N.D.Francis, T.J.Atherton, D.J.Kerbyson, R.A.Packwood and G.J.Vaudin, "Hierarchical Multiple-SIMD Architecture for Image Analysis", Machine Vision and Applications, 1992.
- 5 K.Hwang, H.M.Alnuweiri, V.K.P.Kumar and D.Kim, "Orthogonal Multiprocessor Sharing Memory with an Enhanced Mesh for Integrated Image Understanding" CVGIP: Image Understanding, Vol. 53(1), pp. 31-45, Jan.1991.
- 6 C.C.Weems, "Architecture Requirements of Image Understanding with Respect to Parallel Processing", Proceedings of the IEEE, Vol. 79(4), pp. 537-547, April 1991.
- 7 J.K.Agarwal and N.Nandhakumar, "On the Computation of Motion from Sequences of images -- A Review", Proceedings of the IEEE, Vol. 76(8), pp. 917-935, Aug. 1988.
- 8 H.H.Nagel, "From Image Sequences Towards Conceptual Descriptions", Image and Vision Computing, Vol. 6(2), May 1988.
- 9 M.Jeng and H.J.Siegel, "A Distributed Management Scheme for Partitionable Parallel Computers", Proc. of the Int. Conf. on Parallel Processing, Vol. II, pp. 57-64, Pennsylvania, 1989.
- 10 K.Li and K.H.Cheng, "Job Scheduling in Partitionable Mesh Connected Systems", Proc. of the Int. Conf. on Parallel Processing, Vol. II, pp. 65-72, Pennsylvania, 1989.
- 11 T.J.Patterson, D.M.Chabries and R.W.Christiansen, "Detection Algorithms for Image Sequence Analysis", IEEE Trans. on ASSP, Vol. 37(9), Sept. 1989.
- 12 Y.Z.Hsuel., "New Likelihood Test Methods for Change Detection in Image Sequences", CVGIP, Vol. 26, pp. 73-126, 1984.
- 13 I.K.Sethi and R.Jain, "Finding Trajectories of Feature Points in a Monocular Image Sequence", IEEE Trans. on PAMI, Vol. 9, pp. 56-73, 1987.